On the Synthesis of Stochastic Flow Networks

Hongchao Zhou, Ho-Lin Chen, and Jehoshua Bruck, *Fellow, IEEE*

***Abstract*—A stochastic flow network is a directed graph with incoming edges (inputs) and outgoing edges (outputs), tokens enter through the input edges, travel stochastically in the network and can exit the network through the output edges. Each node in the network is a splitter, namely, a token can enter a node through an incoming edge and exit on one of the output edges according to a predefined probability distribution. We address the following synthesis question: Given a finite set of possible splitters and an arbitrary rational probability distribution, design a stochastic flow network, such that every token that enters the input edge will exit the outputs with the prescribed probability distribution.**

The problem of probability synthesis dates back to von Neumann's 1951 work and was followed, among others, by Knuth and Yao in 1976, who demonstrated that arbitrary rational probabilities can be generated with tree networks; where *minimizing the expected path length*, the expected number of coin tosses in their paradigm, is the key consideration. Motivated by the synthesis of stochastic DNA based molecular systems, we focus on designing *optimal size* stochastic flow networks (the size of a network is the number of splitters). We assume that each splitter has two outgoing edges and is unbiased (probability $\frac{1}{2}$ per output edge). We show that an arbitrary rational probability $\frac{a}{b}$ with $a \leq b \leq 2^n$ can be realized by a stochastic flow network of size n, we also show that this is optimal. We note that our stochastic flow networks have feedback (cycles in the network), in fact, we demonstrate that feedback improves the expressibility of stochastic flow networks, since without feedback only probabilities of the form $\frac{a}{2^n}$ (a an integer) can be realized.

***Index Terms*—Stochastic Flow Network, Graph, Feedback, Probability Synthesis.**

I. Introduction

The problem of probability synthesis dates back to von Neumann's 1951 work [1], where he considered the problem of simulating an unbiased coin by using a biased coin with unknown probability. He noticed that when a coin is tossed twice, the events HT (Heads and then Tail) and TH (Tail and then Heads) have identical probabilities, hence, in his simulation algorithm HT produces the output 0 and TH produces the output 1. The other two events, namely HH and TT, are ignored. Knuth and Yao [2] gave a procedure to generate an arbitrary probability distribution using an unbiased coin. They use the concept of an edge-labeled tree called generating tree and show that the expected number of coin tosses is upper-bounded by the entropy of the target distribution plus two.

In this paper we generalize the concept of a generating tree and consider general directed graphs. Specifically, we introduce the concept of a stochastic flow network - it is a directed graph with incoming edges (inputs) and outgoing edges (outputs), tokens enter through the input edges, travel stochastically in the network and can exit the network through the output edges. Each node in the network is a splitter, namely, a token can enter a node through an incoming edge and exit on one of the output edges according to a predefined probability distribution. Our key motivation is the synthesis of stochastic DNA based molecular systems [3], where splitters are implemented by stochastic chemical reactions, and our goal is to design *optimal size* stochastic flow networks (the size of a network is the number of splitters).

Fig. 1 depicts von Neumann's algorithm in the language of stochastic flow networks. Every splitter corresponds to a coin toss and the probabilities of the H and T edges are p and $(1-p)$, respectively (the value of p is not known). A notation: A splitter with two outgoing edges, with probabilities p and $(1-p)$ will be called a p-splitter.

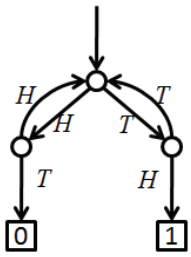
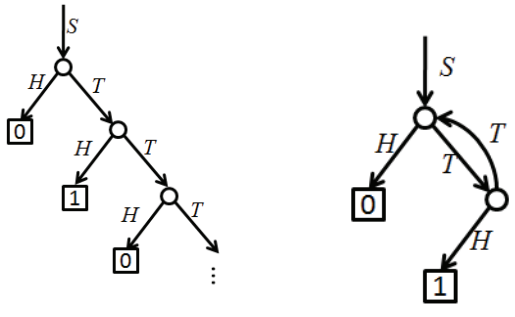


Fig. 1. A network that realizes distribution $\{\frac{1}{2}, \frac{1}{2}\}$, with two p-splitters, where p is unknown.

Fig. 2(a) depicts a generating tree that describes an algorithm for simulating a biased coin with probability $\frac{1}{3}$ using an unbiased coin (probability $\frac{1}{2}$). Each node is a splitter. Assume that a token starts flowing from the root of the tree, at each splitter, it stochastically selects one edge (H with probability p_H or T with probability p_T) to follow. Finally, the token will reach one of the leaves of the tree, called outputs. In



(a) (b)

Fig. 2. (a) An instance of generating tree, which simulates a biased coin with probability $\frac{1}{3}$ using an unbiased coin. (b) An equivalent stochastic flow network.

Hongchao Zhou and Jehoshua Bruck are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA, e-mail: hzhou@caltech.edu; bruck@caltech.edu.

Ho-lin Chen is with the Center for Mathematics of Information, California Institute of Technology, Pasadena, CA 91125, USA, e-mail: holinc@gmail.com.

This work was supported in part by the NSF Expeditions in Computing Program under grant CCF-0832824.

the generating tree of Fig. 2(a), we have only two outputs $\{\beta_1, \beta_2\} = \{0, 1\}$ (corresponding to the labels 0 and 1, respectively). In this example we get that the probability for a token to reach output 0 is $q = \frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^5} + ... = \frac{2}{3}$. Hence the output probability distribution of this tree is $\{\frac{2}{3}, \frac{1}{3}\}$. In general, the outputs of a stochastic flow network have labels denoted by $\{\beta_1, \beta_2, ..., \beta_m\}$. A token will reach an output β_k $(1 \leq k \leq m)$ with probability q_k, and we call $\{q_1, q_2, ..., q_m\}$ the output probability distribution of the network, where $\sum_{k=1}^{m} q_k = 1$.

Now a simple and important observation: The number of splitters in Fig. 2(a) is infinite, however, the equivalent construction of a stochastic flow network in Fig. 2(b) has 2 splitters. The work of Knuth and Yao reasons a generating tree as an algorithm that is maximizing the expected number of desired random bits generated per coin toss. However, motivated by the synthesis of stochastic DNA based molecular systems, we focus on designing *optimal size* stochastic flow networks. This goal is different then the goal in the related literature: Elias [4] demonstrated a construction in which the expected number of unbiased random bits generated per coin toss is asymptotically equal to the entropy of the biased coin. Pae and Loui [5] further proved that the mapping function used by Elias is optimal among all n-randomizing functions and is computable in polynomial time. Han and Hoshi [6] and Abrahams [7] considered the case when the tossed coin is a general biased M-sided coin. Blum [8] have studied a general situation that simulating an unbiased coin using sequences produced by an unknown Markov chain. Gill [9] discussed the problem of generating rational probabilities using a sequential state machine, however, the state machine needs to run for an infinitely long time to get an accurate desired probability. Wilhelm and Bruck [10] proposed a procedure for synthesizing stochastic relay circuits to realized desired binary probabilities. Inspired by PCMOS technology, Qian and Riedel [11] considered the synthesis of of decimal probabilities using combinational logic. However, non of the foregoing approaches considered the problem of generating arbitrary rational probabilities, using a token based approach, while optimizing the network size.

In this paper, we address the following synthesis question: Given a finite set of possible splitters and an arbitrary rational probability distribution, design a stochastic flow network, such that every token that enters the input edge will exit the outputs with the prescribed probability distribution. We assume, without loss of generality, that the probability of each splitter is $\frac{1}{2}$ (since von Neumann's construction in Fig. 1 can use any p-splitter to simulate a $\frac{1}{2}$-splitter). Our goal is to realize the desired probabilities or distributions by constructing a network of minimal size. In addition, we study the expected latency, namely the expected number of splitters a token need to pass before reaching the output.

The main contributions of the paper are

1) *General optimal construction:* For any desired rational probability, an *optimal* size construction of stochastic flow network is provided.
2) *The power of feedback:* We show that with feedback (loops), stochastic flow networks can generate much more probabilities than those without feedback.
3) *Constructions with well-bounded expected latency:* Two additional constructions with a few more splitters than the optimal one are given, such that their expected latencies are well-bounded by constants.
4) *Constructions for arbitrary rational distributions:* We generalize our constructions and results to arbitrary rational probability distributions $\{q_1, q_2, ..., q_m\}$.

The remainder of this paper is organized as follows. In Section II we show how using absorbing Markov chains or Mason's Rule, we can calculate the probability distribution of a given stochastic flow network. Section III introduces our optimal construction for synthesizing stochastic flow networks for arbitrary rational fractions and discusses the power of feedback. Section IV analyzes the expected latency of the optimal construction. Section V gives two constructions with constant-bounded expected latencies, while they still have good performance in network size. Section VI presents a generalization of our results to arbitrary rational probability distributions.

II. Mathematical Tools

In this section, we describe how using absorbing Markov chains or Mason's Rule, we can calculate the probability distribution of a given stochastic flow network.

A. Absorbing Markov Chain

Let's consider a stochastic flow network with n splitters and m outputs, in which each splitter is associated with a state number in $\{1, 2, ..., n\}$ and each output is associated with a state number in $\{n+1, n+2, ..., n+m\}$. When a token is flowing at splitter i with $1 \leq i \leq n$, we say that the current state of this network is i. When it reaches output k with $1 \leq k \leq m$, we say that the current state of this network is $n+k$. Note that the current state of the network only depends on the last state, and when the token reach one output it will stay there for ever. So we can describe token flow in this network using an absorbing Markov chain. If the current state of the network is i, then the probability of reaching state j in the next instant of time is given by p_{ij}. Here, $p_{ij} = p_H$ ($p_{ij} = p_T$) if and only if state i and state j is connected by an edge H (T).

Clearly, we have

$$\begin{array}{ll} \sum_{j=1}^{n+m} p_{ij} = 1 & i = 1, 2, ..., n+m \\ p_{ij} = 0 & \forall i > n \text{ and } i \neq j \\ p_{ii} = 1 & \forall i > n \end{array}$$

Then the network with n splitters and m outputs with different labels can be described by an absorbing Markov chain, where the first n states are transient states and the last m states are absorbing states. The transition matrix of this Markov chain is given by

$$P = \begin{array}{c} \\ n \\ m \end{array} \begin{array}{c} \begin{array}{cc} n & m \end{array} \\ \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \end{array}$$

where Q is an $n \times n$ matrix, R is an $n \times m$ matrix, 0 is an $m \times n$ zeros matrix and I is an $m \times m$ identity matrix.

Let B_{ij} be the probability that an absorbing chain will be absorbed in the absorbing state $j+n$ if it starts in the transient state i. Then B is an $n \times m$ matrix, and

$$B = (I - Q)^{-1}R$$

Assume this Markov chain starts from state 1 and let S_j be the probability that it will be absorbed in the absorbing state $j + n$. Then S is the distribution of the network

$$S = [1, 0, ..., 0]B = e_1(I - Q)^{-1}R$$

In order to make sure that for any incoming tokens, they can reach outputs with probability 1, the transition matrix P of this absorbing Markov chain should satisfy the following condition: For any subset $S \subseteq \{1, 2, ..., n\}$, let P_S denote the square matrix of order $|S|$ obtained from P by selecting the rows and columns in S. Then there exists a row i in matrix P_S such that $\sum_{j \in S} P_{S_{ij}} < 1$. That means that given any subset of transient states, a token can get out of the states in this subset eventually.

Given a stochastic flow network, we can use the formula above to calculate its probability distribution. For example, the transition matrix of the network in Fig. 2(b) is

$$P = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

From which we can obtain the probability distribution of that network

$$S = e_1(I - Q)^{-1}R = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

B. Mason's Rule

Mason's gain rule is a method used in control theory to find the transfer function of a given control system. It can be applied to any signal flow graph. Generally, it can be described as follows (see more details about Mason's rule in [12]):

Let $H(z)$ denote the transfer function of a signal flow graph. Define the following notations:

1) $\Delta(z) =$ determinant of the graph.
2) $L =$ number of forward paths, with $P_k(z)$, $1 \leq k \leq L$ denoting the forward path gains.
3) $\Delta_k(z) =$ determinant of the graph that remains after deleting the kth forward path $P_k(z)$.

Then the transfer function is

$$H(z) = \frac{\sum_{k=1}^{L} P_k(z)\Delta_k(z)}{\Delta(z)} \quad \text{(Mason's formula)}$$

Let's treat stochastic flow network as a control system with input $U(z) = 1$. Applying Mason's rule to this system, we can get the probability P_k that one token will reach output k with $1 \leq k \leq m$. Also having the network in Fig. 2(b) as an example: In this network, we want to calculate the probability for a token to reach output 1 (for short, we call it as the probability of output 1). Since there is only one loop with gain $= \frac{1}{4}$ and only one forward path with forward gain $\frac{1}{4}$, we can obtain that that probability is

$$P = \frac{\frac{1}{4}}{1 - \frac{1}{4}} = \frac{1}{3}$$

which accords with the result from absorbing Markov chain. In fact, it can be proved that the Mason's rule and the matrix form used in absorbing Markov chain are equivalent.

III. Optimal Construction with Feedback

In this section we present an optimal construction of stochastic flow networks, consisting of splitters with probability 1/2, that compute arbitrary rational probabilities. Our constructions have feedback (loops), in fact, we demonstrate that feedback in stochastic flow networks greatly enhance their expressive power.

A. Loop-free networks

Here, we want to study the expressive power of loop-free networks. We say that there are no loops in a network, that means a token can pass any position in the network once or less. For loop-free networks, we have the following theorem:

Theorem 1. *For a loop-free network with n $\frac{1}{2}$-splitters, all probability $\frac{x}{2^n}$ with integer $x(0 \leq x \leq 2^n)$ can be realized, and only probability $\frac{x}{2^n}$ with integer $x(0 \leq x \leq 2^n)$ can be realized.*

Proof: a) In order to prove that all probability $\frac{x}{2^n}$ with integer $x(0 \leq x \leq 2^n)$ can be realized, we only need to provide a procedure to construct a network with size n such that whose probability is $\frac{x}{2^n}$, for an arbitrary integer $x(0 \leq x \leq 2^n)$.

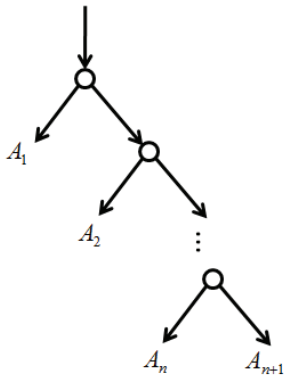


Fig. 3. Tree structure used to realize probability $\frac{x}{2^n}$ for an integer $x(0 \leq x \leq 2^n)$.

1) Construct a tree, as shown in Fig. 3. In this tree structure, each token will reach $A_i(1 \leq i \leq n)$ with probability 2^{-u}, and reach A_{n+1} with probability 2^{-n}.
2) Let $x = \sum_{i=0}^{n-1} \gamma_i 2^i$, where $\gamma_i = 0$ or 1. For each j with $1 \leq j \leq n$, if $\gamma_{n-j} = 1$ we connect A_j to output 0; otherwise, we connect A_j to output 1. Finally, we

connect A_{n+1} to output 1. So far, the probability for a token to reach output 0 is

$$P = \sum_{j=1}^{n} \frac{\gamma_{n-j}}{2^j} = \sum_{i=0}^{n-1} \frac{\gamma_i}{2^{n-i}} = \frac{x}{2^n}$$

Using the procedure above, we can construct a network such that its probability is $\frac{x}{2^n}$.

b) Now, we prove that only probability $\frac{x}{2^n}$ with integer $x(0 \le x \le 2^n)$ can be realized. Namely, in the construction procedure above, the network size n is optimal.

According to Mason's rule, for a network without loops, the probability for a token reaching one output is

$$P = \sum_k P_k$$

where P_k is the path gain of a forward path from the root to the output. Given n splitters, the length of each forward path should be at most n. Otherwise, there must be a loop along this forward path (have to pass the same splitter for at least two times). So for each k, P_k can be written as $\frac{x_k}{2^n}$ for some x_k. As a result, we can get that P can be written as $\frac{x}{2^n}$ for some x. ■

B. Networks with loops

We will show that feedback (loops) can play an important rule to enhance the expressibility of flow networks. For any desired rational probability $\frac{a}{b}$ with integers $0 \le a \le b \le 2^n$, we have the following theorem:

Theorem 2. *For a network with n $\frac{1}{2}$-splitters, all rational probability $\frac{a}{b}$ with integers $0 \le a \le b \le 2^n$ can be realized , and only rational probability $\frac{a}{b}$ with integers $0 \le a \le b \le 2^n$ can be realized.*

Proof: a) We prove that all rational probability $\frac{a}{b}$ with integers $0 \le a \le b \le 2^n$ can be realized. When $b = 2^n$, the problem becomes trivial due to the result of Theorem 1. In the following proof, without loss of generality (w.l.o.g), we only consider the case that $2^{n-1} < b < 2^n$ for some n.

In order to prove this, we first prove that all probability distributions $\{\frac{x}{2^n}, \frac{y}{2^n}, \frac{z}{2^n}\}$ with integers x, y, z s.t. $(x+y+z = 2^n)$ can be realized with n splitters. Now we prove this by induction on n, by constructing the corresponding network iteratively.

When $n = 1$, by enumerating all the possible connections, the following probability distributions can be realized:

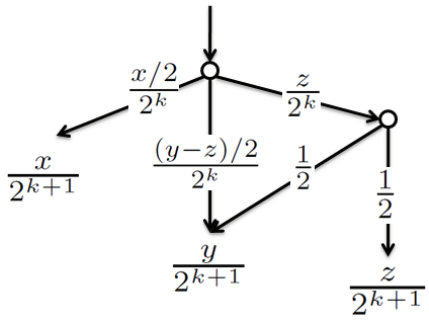
$$\{0,0,1\}, \{0,1,0\}, \{1,0,0\}, \{0,\frac{1}{2},\frac{1}{2}\}, \{\frac{1}{2},0,\frac{1}{2}\}, \{\frac{1}{2},\frac{1}{2},0\}$$

So all probability distributions $\{\frac{x}{2}, \frac{y}{2}, \frac{z}{2}\}$ with integers x, y, z s.t. $(x + y + z = 2)$ can be realized.

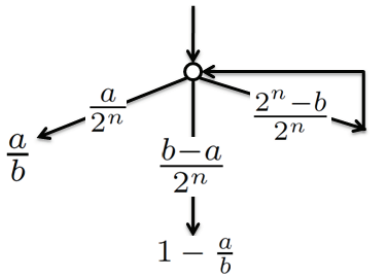
Assume that all probability distribution $\{\frac{x}{2^k}, \frac{y}{2^k}, \frac{z}{2^k}\}$ with integers x, y, z s.t. $(x + y + z = 2^k)$ can be realized by a network with k splitters. Then we show that any desired probability distribution $\{\frac{x}{2^{k+1}}, \frac{y}{2^{k+1}}, \frac{z}{2^{k+1}}\}$ s.t. $x + y + z = 2^{k+1}$ can be realized with one more splitter. Since $x+y+z = 2^{k+1}$, we know that at least one of x, y, z is even. W.l.o.g, we let x be even. Then either both y and z are even, or both y and z are odd.

When both y and z are even, the problem is trivial since the desired probability distribution can be written as $\{\frac{x/2}{2^k}, \frac{y/2}{2^k}, \frac{z/2}{2^k}\}$, which can be realized by a network with k splitters.

When both y and z are odd, W.l.o.g, we assume that $z \le y$. In this case, we construct a network to realize probability distribution $\{\frac{x/2}{2^k}, \frac{(y-z)/2}{2^k}, \frac{z}{2^k}\}$ with k splitters. By connecting the last output with probability $\frac{z}{2^k}$ to an additional splitter, we can get a new network in Fig. 4(a), whose probability distribution is $\{\frac{x}{2^{k+1}}, \frac{y}{2^{k+1}}, \frac{z}{2^{k+1}}\}$.



(a)



(b)

Fig. 4. (a) The network to realize $\{\frac{x}{2^{k+1}}, \frac{y}{2^{k+1}}, \frac{z}{2^{k+1}}\}$ iteratively. (b) The network to realize $\{\frac{a}{b}, 1 - \frac{a}{b}\}$.

Hence, for any probability distribution $\{\frac{x}{2^n}, \frac{y}{2^n}, \frac{z}{2^n}\}$ with $x + y + z = 2^n$, we can always construct a network with n splitters to realize it.

Now, in order to realize probability $\frac{a}{b}$ with $2^{n-1} < b < 2^n$ for some n, we can construct a network with probability distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$ with n splitters and connect the last output (output 2) to the starting point of the network, as shown in Fig. 4(b). Using the method in Section II, we can compute that the probability for a token to reach output 0 is $\frac{a}{b}$. A simple understanding for this result is that: (1) the ratio of the probabilities for a token to reach the first output and the second output is $\frac{a}{2^n} : \frac{b-a}{2^n}$ (2) the sum of these probabilities is 1, since tokens will finally reach one of the two outputs.

b) Now we prove that with n splitters, only rational probability $\frac{a}{b}$ with integers $0 \le a \le b \le 2^n$ can be realized. For any flow network with n splitters to generate a probability, it can be described by an absorbing Markov chain with n transient states and 2 absorbing states, whose transition matrix P can

be written as

$$P = \begin{pmatrix} p_{11} & \dots & p_{1n} & p_{1(n+1)} & p_{1(n+2)} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ p_{n1} & \dots & p_{nn} & p_{n(n+1)} & p_{n(n+2)} \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

where each row consists of two $\frac{1}{2}$ entries and n zeros entries.

Let

$$Q = \begin{pmatrix} p_{11} & \dots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \dots & p_{nn} \end{pmatrix}, R = \begin{pmatrix} p_{1(n+1)} & p_{1(n+2)} \\ \vdots & \vdots \\ p_{n(n+1)} & p_{n(n+2)} \end{pmatrix}$$

then the probability distribution of the network can be written as

$$e_1(I-Q)^{-1}R$$

In order to prove the result in the theorem, we only need to prove that $(I-Q)^{-1}R$ can be written as $\frac{1}{b}A$ with $b \leq 2^n$, where A is an integer matrix.

Let $K = I - Q$, we know that K is invertible if and only $det(K) \neq 0$. In this case, we have

$$(K^{-1})_{ij} = \frac{K_{ji}}{det(K)}$$

where K_{ji} is defined as the determinant of the square matrix of order $(n-1)$ obtained from K by removing the i^{th} row and the j^{th} column multiplied by $(-1)^{i+j}$.

Since each entry of K is chosen from $\{0, \frac{1}{2}, 1\}$, K_{ji} can be written as $\frac{k_{ji}}{2^{n-1}}$ for some integer k_{ji} and $det(K)$ can be written as $\frac{b}{2^n}$ for some integer b. According to Lemma 1 in the appendix, we have $0 \leq det(K) \leq 1$, which leads us to $0 < b \leq 2^n$ (note that $det(K) \neq 0$).

Then, we have that

$$\begin{aligned} K^{-1} &= \frac{1}{det(K)} \begin{pmatrix} K_{11} & K_{21} & \dots & K_{n1} \\ K_{12} & K_{22} & \dots & K_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ K_{1n} & K_{2n} & \dots & K_{nn} \end{pmatrix} \\ &= \frac{2}{b} \begin{pmatrix} k_{11} & k_{21} & \dots & k_{n1} \\ k_{12} & k_{22} & \dots & k_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ k_{1n} & k_{2n} & \dots & k_{nn} \end{pmatrix} \end{aligned}$$

Since each entry of R is also in $\{0, \frac{1}{2}, 1\}$, we know that

$$2R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ \vdots & \vdots \\ r_{n1} & r_{n2} \end{pmatrix}$$

is an integer matrix.

As a result

$$\begin{aligned} K^{-1}R &= \frac{2R}{b} \begin{pmatrix} k_{11} & k_{21} & \dots & k_{n1} \\ k_{12} & k_{22} & \dots & k_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ k_{1n} & k_{2n} & \dots & k_{nn} \end{pmatrix} \\ &= \frac{1}{b} \begin{pmatrix} k_{11} & k_{21} & \dots & k_{n1} \\ k_{12} & k_{22} & \dots & k_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ k_{1n} & k_{2n} & \dots & k_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ \vdots & \vdots \\ r_{n1} & r_{n2} \end{pmatrix} \\ &= \frac{A}{b} \end{aligned}$$

where each entry of A is an integer.

This completes the proof. ■

Based on the method in the theorem above, we can realize any arbitrary rational probability with an optimal size network. The construction has two steps:

1) Construct a network with output distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$ iteratively, using the method in Theorem 2, with at most n splitters.
2) Connect the last output to the starting point (feedback), such that the distribution of the new network is $\{\frac{a}{b}, \frac{b-a}{b}\}$.

Note that when $b = 2^n$ for some n, the construction above is the same as that in Theorem 1. Now, assume we want to realize probability $\frac{14}{29}$. We can first generate a probability distribution $\{\frac{14}{32}, \frac{15}{32}, \frac{3}{32}\}$, which can be realized by adding one splitter to a network with probability distribution $\{\frac{7}{16}, \frac{6}{16}, \frac{3}{16}\}$... Iteratively, we can get the following probability distributions:

$$\begin{aligned} \{\frac{14}{32}, \frac{15}{32}, \frac{3}{32}\} \rightarrow \{\frac{7}{16}, \frac{6}{16}, \frac{3}{16}\} \rightarrow \{\frac{2}{8}, \frac{3}{8}, \frac{3}{8}\} \\ \rightarrow \{\frac{1}{4}, 0, \frac{3}{4}\} \rightarrow \{\frac{1}{2}, 0, \frac{1}{2}\} \end{aligned}$$

Hence, we can get a network to generate probability distribution $\{\frac{14}{32}, \frac{15}{32}, \frac{3}{32}\}$, as shown in Fig. 5(a), where only 5 splitters are used. After connecting the last output to the starting point, we can get the network in Fig. 5(b) with probability $\frac{14}{29}$. Comparing the results in Theorem 2 with those in Theorem 1, we can see that introducing loops into networks can strongly enhance the expressibility of the network.

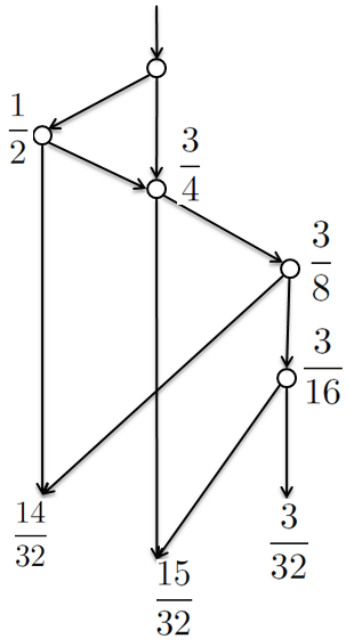
IV. Expected latency of Optimal Construction

In this section, we consider the expected latency, defined as the expected number of splitters a token need to pass before reaching one of the outputs. For the optimal construction proposed above, called Scheme A, we have the following theorems about its expected latency.

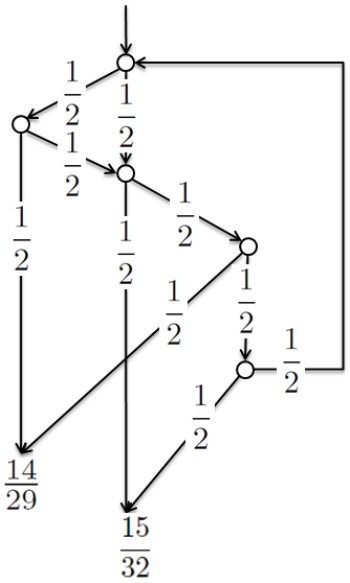
Theorem 3. *Given a network with rational probability $\frac{a}{b}$ with $b \leq 2^n$ constructed using the optimal construction (Scheme A), its expected latency ET is upper bounded by* [1]

$$ET \leq (\frac{3n}{4} + \frac{1}{4})\frac{2^n}{b} < \frac{3n}{2} + \frac{1}{2}$$

[1]By making scheme A more sophisticated, we can reduce the upper bound to $(\frac{n}{2} + \frac{3}{4})\frac{2^n}{b}$.



(a)



(b)

Fig. 5. (a) The network to realize probability distribution $\{\frac{14}{32}, \frac{15}{32}, \frac{3}{32}\}$ (b)The network to realize probability $\frac{14}{29}$.

Proof: For scheme A, we first prove that the expected latency of the network with distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$ is bounded by $\frac{3n}{4} + \frac{1}{4}$.

Let's prove this by induction. When $n = 0$ or $n = 1$, this conclusion is true. Assume when $n = k$, this conclusion is true, we want to show that the conclusion is also true for $n = k+2$. Note that in scheme A, a network with size $k+2$ and three outputs can be constructed by adding two more splitters to a network with size k. Let T_k denote the latency of the network with size k, then

$$E[T_{k+2}] = E[T_k] + p_1 + p_2$$

where p_1 is the probability for a token to reach the first additional splitter and p_2 is the probability for a token to reach the second additional splitter. Assume the distribution of the network with size k is $\{q_1, q_2, q_3\}$, then

$$p_1 + p_2 \leq \max_{i \neq j}(q_i + (\frac{q_i}{2} + q_j)) \leq \frac{3}{2}$$

So the conclusion is true for $n = k + 2$. By induction, we know that it holds for all $n \in \{0, 1, 2, ...\}$.

Secondly, we prove that if the expected latency of the network with distribution $\{q_1, q_2, q_3\}$ is ET', then by connecting its last output to its starting point (feedback), we can get a network such that its expected latency is $ET = \frac{ET'}{q_1+q_2}$. This conclusion can be obtained immediately from

$$ET' = ET + q_3(ET')$$

The theorem holds based on the two conclusions above. ■

Theorem 4. *For any network size n, there exists a network constructed using the optimal construction (Scheme A) such that its expected latency ET is lower bounded by*

$$ET \geq \frac{n}{3} + \frac{2}{3}$$

Proof: We only need to construct a network with distribution $\{\frac{x}{2^n}, \frac{y}{2^n}, \frac{z}{2^n}\}$ for some integers x, y, z such that its expected latency is lower bounded by $\frac{n}{3} + \frac{2}{3}$.

Let's construct such a network in the following way: Starting from a network with single splitter, and at each step adding one more splitter. Assume the current distribution is $\{p_x, p_y, p_z\}$ with $p_x \geq p_y \geq p_z$ (if this is not true, we can change the order of the outputs), then we can add an additional splitter as shown in Fig. 6. Iteratively, with n splitters, we can construct a network with distribution $\{\frac{x}{2^n}, \frac{y}{2^n}, \frac{z}{2^n}\}$ for some integers x, y, z and its expected latency is more than $\frac{n}{3} + \frac{2}{3}$ (This can be proved by induction).

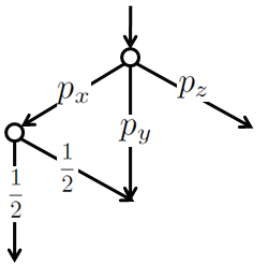


Fig. 6. Illustration for the construction of a network with unbounded expected latency. Here, we have $p_x \geq p_y \geq p_z$.

By connecting one output with probability smaller than $\frac{1}{2}$ to the starting point, we can get such a network. ■

The theorems above show that the upper bound of the expected latency for the optimal construction is not well-bounded. However, this upper bound only reflects the worst case. That does not mean that the optimal construction always has a bad performance in expected latency when network size is large. Let's consider the case that the desired probability is $\frac{a}{b}$ with $b = 2^n$ for some n. In this case, the optimal method

	Scheme A	Scheme B	Scheme C
Network size	$\leq n$	$\leq n+3$	$\leq 2(n-1)$
Expected latency	$\leq (\frac{3n}{4}+\frac{1}{4})\frac{2^n}{b}$	$\leq 6\frac{2^n}{b}$	$\leq 3.585\frac{2^n}{b}$

TABLE I
THE COMPARISON OF DIFFERENT SCHEMES, HERE $\frac{2^n}{b} < 2$.

above constructs the same network with that in Theorem 1, whose expected latency can be written as

$$\begin{aligned} ET &= \sum_{i=1}^{n} \frac{i}{2^i} + \frac{n}{2^n} \\ &= [\sum_{i=1}^{n} x^{i+1}]' - \sum_{i=1}^{n-1} \frac{i}{2^i} \\ &= [\frac{x^2 - x^{n+2}}{1-x}]' - \frac{x - x^n}{1-x} \\ &= 2 - \frac{1}{2^{n-1}} \end{aligned}$$

which is well-bounded by 2.

V. Alternative Constructions

In the last section, we show that the expected latency of the optimal construction (Scheme A) is not always well-bounded. In this section, we give two other constructions (Scheme B and Scheme C) such that their expected latencies are well-bounded, while their network size is close to optimality. Table I shows the summary of the results in this section, from which we can see that there is a tradeoff between the upper-bound on the network size and the upper-bound on the expected latency.

A. Scheme B

Assume that the desired probability is $\frac{a}{b}$ with $2^{n-1} < b \leq 2^n$ for some n. In this subsection, we give a construction (scheme B) with at most $n+3$ splitters such that its expected latency is well-bounded by a constant.

Assume a and b are relatively prime, and let $c = b - a$. Then $\frac{a}{2^n}$ and $\frac{c}{2^n}$ can be expressed using binary extension.

$$\frac{a}{2^n} = \sum_{i=1}^{n} a_i 2^{-i}$$

$$\frac{c}{2^n} = \frac{b-a}{2^n} = \sum_{i=1}^{n} c_i 2^{-i}$$

Starting from the structure in Fig. 7, we connect A_i with $1 \leq i \leq n+1$ to one of B_1, B_2, B_3 and output 2, such that the probability distribution of the outputs is $\{\frac{a}{2^{n+1}}, \frac{b-a}{2^{n+1}}, \frac{2^{n+1}-b}{2^{n+1}}\}$. Based on the values of a_i, c_i with $0 \leq i \leq n-1$, we have the following rules for these connections:

1) If $a_i = c_i = 1$, connect A_i with B_1.
2) If $a_i = 1, c_i = 0$,connect A_i with B_2.
3) If $a_i = 0, c_i = 1$, connect A_i with B_3.
4) If $a_i = c_i = 0$, connect A_i with output 2.

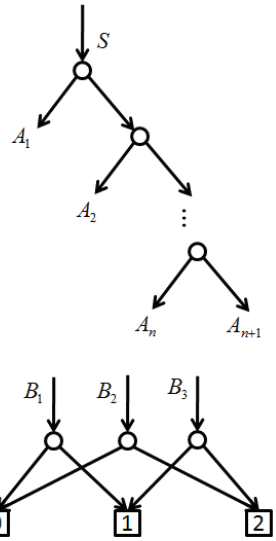


Fig. 7. The framework to realize probability $\frac{a}{b}$.

5) Connect A_{n+1} with output 2.

Assume that the probability for a token to reach B_j with $1 \leq j \leq 3$ is $P(B_j)$, then we have

$$P(B_1) = \sum_{i=1}^{n} I_{(a_i=c_i=1)} 2^{-i}$$

$$P(B_2) = \sum_{i=1}^{n} I_{(a_i=1,c_i=0)} 2^{-i}$$

$$P(B_3) = \sum_{i=1}^{n} I_{(a_i=0,c_i=1)} 2^{-i}$$

where $I_\phi = 1$ if and only if ϕ is true, otherwise $I_\phi = 0$.

As a result, the probability for a token to reach the first output is

$$P_1 = \frac{1}{2}(P(B_1) + P(B_2)) = \frac{1}{2}\sum_{i=1}^{n} I_{(a_i=1)} 2^{-i} = \frac{a}{2^{n+1}}$$

Similarly, the probability for a token to reach the second output is

$$P_2 = \frac{b-a}{2^{n+1}}$$

So far, we get that the distribution of the network is $\{\frac{a}{2^{n+1}}, \frac{b-a}{2^{n+1}}, \frac{2^{n+1}-b}{2^{n+1}}\}$. Similar as Theorem 2, by connecting the output 2 to the starting point (feedback), we can get a new network with probability $\frac{a}{b}$. Note that comparing with the optimal scheme, 3 more splitters are used to realize the desired probability. For this network, we compute an upper bound on its expected latency:

Theorem 5. *Given a network with probability $\frac{a}{b}$ ($2^{n-1} < b < 2^n$) constructed using scheme B, its expected latency ET is bounded by*

$$ET \leq 6\frac{2^n}{b} < 12$$

Proof: First, without the feedback, the expected latency for a token to reach B_1, B_2, B_3 or output 2 is less than 2.

This can be obtained from the example in the last section. As a result, without the feedback, the expected latency for a token to reach one of the outputs is less than 3. Finally, we can get the theorem. ■

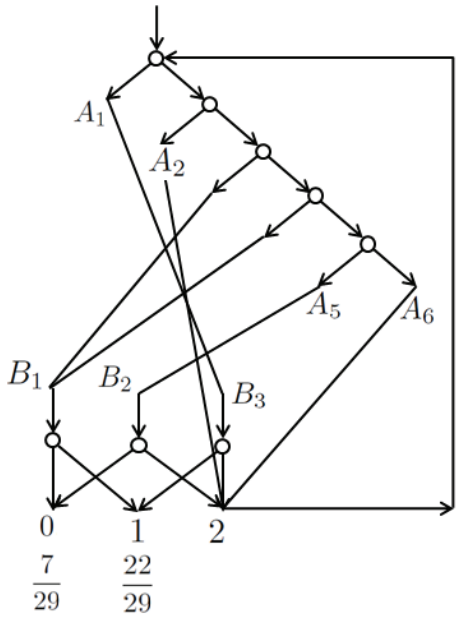


Fig. 8. The network to realize probability $\frac{7}{29}$.

Let's give an example of scheme B. Assume the desired probability is $\frac{7}{29}$, then we can write $\frac{a}{2^n}$ and $\frac{b-a}{2^n}$ into binary expansions:

$$\begin{aligned} \frac{a}{2^n} &= 0.00111 \\ \frac{b-a}{2^n} &= 0.10110 \end{aligned}$$

According to the rules above, we connect A_1 to B_3, A_2 to output 2,... After connecting output 2 to the starting point, we can get a network with probability $\frac{7}{29}$, as shown in Fig. 8.

Based on Scheme B, we can also construct an Universal Probability Generator (UPG) efficiently with $a_i, c_i (0 \leq i \leq n-1)$ as inputs, such that its probability output is $\frac{a}{a+c} = \frac{a}{b}$. The definition and description of UPG can be found in [10]. Instead of connecting A_i with $1 \leq i \leq n$ to one of B_1, B_2, B_3 and output 2 directly, we insert a deterministic device as shown in Fig. 9. At each node of this device, if its corresponding input is 1, all the incoming tokens will exit the left outgoing edge. If the input is 0, all the incoming tokens will exit the right outgoing edge. As a result, the connections between A_i and B_1,B_2,B_3,Output 2 are automatically controlled by inputs a_i and c_i. Finally, we can get an Universal Probability Generator (UPG), whose output probability is

$$\frac{\sum_{i=0}^{n-1} a_i 2^i}{\sum_{i=0}^{n}(a_i + c_i)2^i}$$

B. Scheme C

In this subsection, we propose another scheme, called scheme C, which is similar to Scheme A. Both Scheme A and

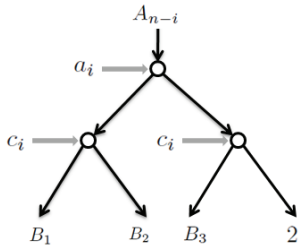


Fig. 9. The deterministic device to control flow in UPG.

Scheme C is try to realize the distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$ first. However, the difference is that in Scheme C, this distribution is realized by applying Knuth and Yao's scheme [2]. Generally, Knuth and Yao's scheme can be described as follows [13]: Assume we want to realize the distribution $\{p_1, p_2, ...\}$. Let the binary expansion of the probability p_i be $p_i = \sum_{j \geq 1} p_i^{(j)}$, where $p_i^{(j)} = 2^{-j}$ or 0. Then the atoms of the expansion are $\{p_i^{(j)} : i = 1, 2, ..., m, j \geq 1\}$.

Since $\sum_i p_i = 1$, the sum of the probabilities of these atoms is 1. Now, we allot all the atoms to leaves of a tree such that the depth of atom 2^{-j} is j. We can see that all the depth of these atoms satisfy the Kraft inequality, and hence we can always construct such a tree.

Knuth and Yao showed that the expected number of fair bits required by the procedure above to generate a random variable X with distribution $\{p_1, p_2, ...\}$ lies between $H(X)$ and $H(X) + 2$. Based on this result, we have the following theorem about Scheme C.

Theorem 6. *Given a network with probability $\frac{a}{b}$ ($2^{n-1} < b < 2^n$) constructed using scheme C, its network size is bounded by $2(n-1)$ and its expected latency ET is bounded by*

$$ET \leq (log_2 3 + 2)\frac{2^n}{b} < 7.2$$

Proof: Let's first consider the network with distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$, which is constructed using Knuth and Yao's scheme.

1) The network size is bounded by $2(n-1)$. That is because for each j with $2 \leq j \leq n$, there are at most two atoms with value 2^{-j}. If $j = 1$, there are at most one atom with value 2^{-j} (except that the target distribution is $\{\frac{1}{2}, \frac{1}{2}\}$).

2) The expected latency ET' of the network with distribution $\{\frac{a}{2^n}, \frac{b-a}{2^n}, \frac{2^n-b}{2^n}\}$ is bounded by $ET' \leq (log_2 3 + 2)$. That is because the expected latency ET' is equal to the expected number of fair bits required. According to the result of Knuth and Yao, it is not hard to get this conclusion.

Now we can get a new network by connecting the last output to the starting point (feedback). We can see that the network size keeps unchanged and the expected latency of the new network is $ET = ET'\frac{2^n}{b}$. ■

Let's go back to the example of realizing probability $\frac{14}{29}$. According to Knuth and Yao's scheme, we need first find the atoms for the binary expansions of $\frac{14}{32}, \frac{15}{32}, \frac{3}{32}$, i.e.

$$\frac{14}{32} \rightarrow (\frac{1}{4}, \frac{1}{8}, \frac{1}{16})$$

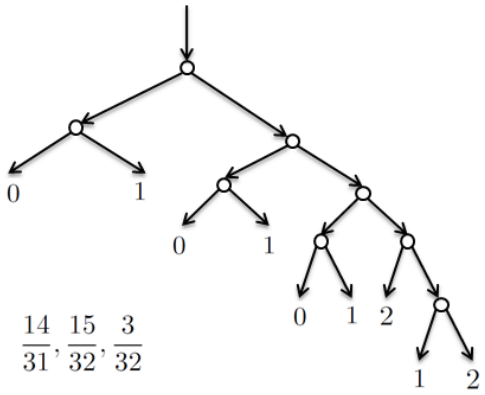


Fig. 10. The network to realize probability distribution $\{\frac{14}{32}, \frac{15}{32}, \frac{3}{32}\}$ using Knuth and Yao's scheme.

$$\frac{15}{32} \to (\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32})$$
$$\frac{3}{32} \to (\frac{1}{16}, \frac{1}{32})$$

Then we allot these atoms to a binary tree, as shown in Fig. 10. In this tree, the probability for a token to reach outputs labeled 0 is $\frac{14}{32}$ and the probability for a token to reach outputs labeled 1 is $\frac{15}{32}$. If we connect the outputs labeled 2 to the starting point, the desired probability $\frac{14}{29}$ can be achieved.

VI. Generating Rational Distributions

In this section, we want to generalize our results to generate arbitrary rational probability distributions $\{q_1, q_2, ..., q_m\}$ with $m \geq 2$. Two different methods will be proposed and studied. The first method is based on Knuth and Yao's scheme and it is a direct generalization of Scheme C. The second method is based on binary-tree structure. At each parent node of the binary tree, one probability is divided into two probabilities. As a result, using a binary-tree structure, the probability one can be divided into m probabilities (as a distribution). In the rest of this section, we will discuss and analyze these two methods, where we write $\{q_1, q_2, ..., q_m\}$ as $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$ with b minimized.

A. Based on Knuth and Yao's scheme

Similar as Scheme C in the section above, in order to generate distribution $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$ with $2^{n-1} < b \leq 2^n$ for some n, we can first construct a network with distribution $\{\frac{a_1}{2^n}, \frac{a_2}{2^n}, ..., \frac{a_m}{2^n}, \frac{2^n-b}{2^n}\}$ using Knuth and Yao's scheme. Then by connecting the last output to the starting point (feedback), we can obtain a network with distribution $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$. In order to study the properties of this method, we will analyze the two extreme cases: (1) $m = b$ and (2) $m \ll b$.

When $m = b$, the target probability distribution can be written as $\{\frac{1}{b}, \frac{1}{b}, ..., \frac{1}{b}\}$. For this distribution, we have the following theorem about the network constructed using the method based on Knuth and Yao's scheme.

Theorem 7. *For a distribution $\{\frac{1}{b}, \frac{1}{b}, ..., \frac{1}{b}\}$, the method based on Knuth and Yao's scheme can construct a network with $b + h(b) - 1$ splitters. Here, we assume $b = 2^n - \sum_{i=0}^{n-1} \gamma_i 2^i$ then $h(b) = \sum_{i=0}^{n-1} \gamma_i$.*

Proof: See the network in Fig. 11 as an example for the construction.

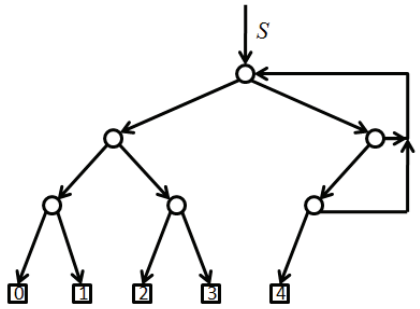


Fig. 11. The network to realize probability distribution $\{\frac{1}{5}, \frac{1}{5}, ..., \frac{1}{5}\}$.

First, let's consider a complete tree with depth n. The network size of such a tree (i.e. the number of parent nodes) is $2^n - 1$, denoted by $N_{complete}$.

Let $N(b)$ be the network size of the construction above to realize distribution $\{\frac{1}{b}, \frac{1}{b}, ..., \frac{1}{b}\}$. Assume

$$2^n - b = 2^{a_1} + 2^{a_2} + ... + 2^{a_H}$$

with $n > a_1 > a_2 > ... > a_H$ is a binary expansion of $2^n - b$, then we can get that the difference between the size of the construction and the size of the complete binary tree is

$$\Delta = N_{complete} - N(b) = \sum_{i=1}^{H} (2^{a_i} - 1) = 2^n - b - H$$

So the network size of the construction $N(b)$ is

$$N(b) = 2^n - 1 - (2^n - b - H) = b + H - 1$$

where $H = \sum_{i=0}^{n-1} \gamma_i = h(b)$. ■

Let $N^*(b)$ be the optimal network size. Note that $h(b)$ is at most the number of bits in the binary expansion of $2^n - b$ (which is smaller than b), so we can get the following inequation quickly

$$b - 1 \leq N^*(b) \leq N(b) \leq b - 1 + \log_2 b$$

It shows that the construction based on Knuth and Yao's scheme is almost optimal (probably optimal) when $m = b$. Further more, we believe that when m is large, this construction has good performance in network size.

For the expected latency, using the same argument in Theorem 6, we can get the following theorem.

Theorem 8. *For a distribution $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$ with $b \leq 2^n$, the method based on Knuth and Yao's scheme can construct a network with at most $m(n - \lfloor log_2 m \rfloor)$ splitters, such that its expected latency ET is bounded by*

$$H(X')\frac{2^n}{b} \leq ET \leq [H(X') + 2]\frac{2^n}{b}$$

where $\frac{2^n}{b} < 2$. $H(X')$ is the entropy of the distribution $\{\frac{a_1}{2^n}, \frac{a_2}{2^n}, ..., \frac{a_m}{2^n}, \frac{2^n-b}{2^n}\}$.

This theorem is a simple generalization of the results in Theorem 6. Here, the upper bound for the network size is only tight for small m.

B. Based on binary-tree structure

Generally, the method based on Knuth and Yao's scheme has good performances in both network size and expected latency. In this subsection, we propose another method to generate an arbitrary rational distribution $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$ such that it may have better performance in network size when m is small enough. The idea of this method is based on binary-tree structure. We can describe the method in the following way: We construct a binary tree with m leaves, where the weight of the ith $(1 \leq i \leq m)$ leaf is $q_i = \frac{a_i}{b}$. For each parent node, its weight is sum of the weights of its two children. Recursively, we can get the weight of the root is 1. Now, for each parent node, assume the weights of its two children are w_1 and w_2, then we can replace this parent node by a subnetwork with probability distribution $\{\frac{w_1}{w_1+w_2}, \frac{w_2}{w_1+w_2}\}$. Finally, we replace each leaf with an output. In this new network, a token will reach the i^{th} output with probability q_i.

For example, in order to realize the distribution $\{\frac{1}{2}, \frac{1}{6}, \frac{1}{4}, \frac{1}{12}\}$, we can first generate a binary-tree with 4 leaves, such as the binary-tree in Fig. 12(a). Then according to the method above, we can obtain the weight of each node in this binary tree, see Fig. 12(b). Based on these weights, we can replace the three parent nodes with three subnetworks, whose probability distributions are $\{\frac{1}{2}, \frac{1}{2}\}, \{\frac{1}{3}, \frac{1}{3}\}, \{\frac{3}{4}, \frac{1}{4}\}$. Eventually, we can construct a network with the desired distribution as shown in Fig. 12(c).

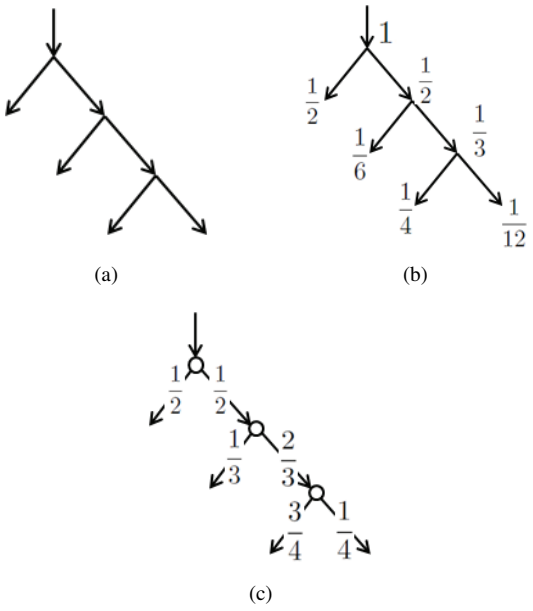


Fig. 12. (a) A binary-tree with 4 leaves. (b) Node weights in the binary tree. (c) The network to realize probability distribution $\{\frac{1}{2}, \frac{1}{6}, \frac{1}{4}, \frac{1}{12}\}$, where $\{\frac{1}{3}, \frac{2}{3}\}, \{\frac{3}{4}, \frac{1}{4}\}$ can be realized using the methods in the sections above.

During the procedure above, any binary-tree with m leaves works. Among all these binary-trees, we need to find one such that the resulting network satisfies our requirements in network size and expected latency. When m is extremely small, such as $3, 4$, we can search all the binary-trees with m leaves. However, when m is a little larger, such as 10, the number of such binary-trees grows exponentially. In the rest of this section, we will show that Huffman procedure can create a binary-tree with good performances in network size and expected latency for most of the cases.

Huffman procedure can be described as follows:

1) Draw m nodes with weights $q_1, q_2, ..., q_m$.
2) Let S denote the set of nodes without parents. Assume node A and node B are the two nodes with the minimal weights in S, then we added a new node as the parent of A and B, with weight $w(A) + w(B)$, where $w(X)$ is the weight of node X.
3) Repeat 2) until the size of S is 1.

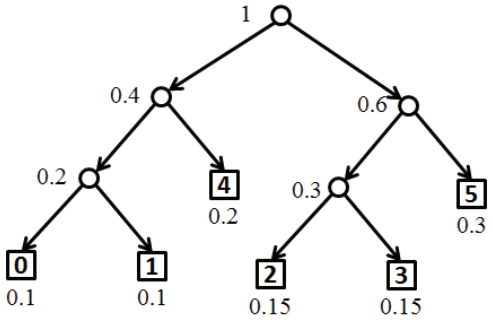


Fig. 13. The tree constructed using Huffman procedure when the desired distribution is $\{0.1, 0.1, 0.15, 0.15, 0.2, 0.3\}$

Fig. 13 shows an example of a binary-tree constructed by Huffman procedure, when the desired distribution is $\{0.1, 0.1, 0.15, 0.15, 0.2, 0.3\}$. From [13], we know that using Huffman procedure, we can create a tree with minimal expected path length. Let EL^* denote this minimal expected path length, then its satisfies the following inequation,

$$H(X) \leq EL^* \leq H(X) + 1$$

where $H(X)$ is the entropy of the desired probability distribution $\{q_1, q_2, ..., q_m\} = \{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$.

Now, in order to simplify our analysis, we assume that (1) the expected latency of each node in the binary-tree is the same, denoted as ET_e, and (2) each node can be replaced with about $\log_2(bw)$ splitters, where w is the weight of the node. These simplifications are reasonable, due to the results about our constructions in the sections above. If these assumptions are true, we can get that the expected latency ET of the constructed network satisfies

$$H(X)ET_e \leq ET \leq (H(X) + 1)ET_e$$

which is optimal among all the binary-tree structures. On the other hand, we consider the size of a network constructed based on binary-tree structure. Let w_i denote the weight of i^{th} parent node in the binary tree, then the network size will be $\sum_{i=1}^{m-1} \log_2(bw_i)$. According to Lemma 2 in the Appendix, we get that when m is small, Huffman procedure can create a binary-tree to minimize $\sum_{i=1}^{m-1} w_i$. As a result, among all the binary-trees with m leaves, Huffman procedure can create one with minimized network size. Note that the conclusions above are completely based on our assumptions. However, these assumptions are not always true. For example, let's consider a desired distribution $\{q_1, q_2, ..., q_m\}$ with $\sum_{i \in S} q_i = \frac{1}{2}$ for

	Based on KY's Scheme	Based on binary-tree
Network size	$\leq m(n - \lfloor \log_2 m \rfloor)$	$\leq (m-1)n$
Expected latency	$\leq (H(X')+2)\frac{2^n}{b}$	$\leq (H(X)+1)ET_{max}$

TABLE II
THE COMPARISON OF DIFFERENT METHODS, HERE $\frac{2^n}{b} < 2$

some set S. In this case, the binary-tree constructed using Huffman procedure may not be the best one.

C. Comparison

Let's have a brief comparison between the method based on Knuth and Yao's scheme and the method based on binary-tree structure. Generally, when m is large, we think that the method based Knuth and Yao's scheme may perform better. When m is very small, the comparison between these two methods is given in Table II, where the desired distribution is $\{\frac{a_1}{b}, \frac{a_2}{b}, ..., \frac{a_m}{b}\}$ with $2^{n-1} < b \leq 2^n$. In this table, we assume that the binary-tree is constructed using Huffman procedure. Let $H(X)$ denote the entropy of this desired distribution and $H(X')$ denote the entropy of distribution $\{\frac{a_1}{2^n}, \frac{a_2}{2^n}, ..., \frac{a_m}{2^n}, \frac{2^n-b}{2^n}\}$. It is not hard to get that $H(X') \leq H(X) + \frac{1}{2}$. ET_{max} denotes the maximal expected latency of the parent nodes in a given binary-tree. Normally, we have $ET_{max} > 2$. We can see that the method based on binary-tree structure has a better (a little) theoretical upper bound for the network size. However, it is still hard to say that one of the two methods has a better performance than the other one, no matter in network size and expected latency, when m is very small.

VII. CONCLUSION

Motivated by computing with stochastic chemical reactions, we introduced the concept of stochastic flow networks and studied the synthesis of minimal size networks for realizing rational probabilities. We also studied the expected latency of stochastic flow networks, namely the expected number of splitters a token need to pass before reaching the output. Two constructions with well-bounded expected latency are proposed. Finally, we generalize our results to arbitrary rational probability distributions.

Beside of network size and expected latency, robustness is also an important issue in stochastic flow networks. Assume the probability error of each splitter is bounded by a constant ϵ, the robustness of a given network can be measured by the total probability error. It can be shown that most constructions in this paper are robust against small errors in the splitters. There are still some open questions about stochastic flow networks: given some splitters with probability p (known or known), how to construct flow networks to generate rational probabilities directly (without generating probability $\frac{1}{2}$)? How to approximate arbitrary probabilities or distributions (irrational)? If different splitters have different probabilities, how can we answer these questions?

APPENDIX

Lemma 1. *Given Q an $n \times n$ matrix with each entry in $\{0, \frac{1}{2}, 1\}$, such that sum of each row is at most 1, then we have $0 \leq det(I - Q) \leq 1$, where I is an identity matrix and $det(\cdot)$ is the determinant of a matrix.*

Proof: Before proving this lemma, we can see that for any given matrix Q, it has the following properties: For any i, j such that $1 \leq i < j \leq n$, switching the i^{th} row with the j^{th} row then switching the i^{th} column with the j^{th} column, the determinant of $K = I - Q$ keeps unchanged. And more, each entry of Q is still from $\{0, \frac{1}{2}, 1\}$ and sum of each row of Q is at most 1. Now, we call the transform above as equivalent transform of Q.

Let's prove this lemma by induction. When $n = 1$, we have that

$$Q = \begin{pmatrix} 0 \end{pmatrix} \text{ or } Q = \begin{pmatrix} \frac{1}{2} \end{pmatrix} \text{ or } Q = \begin{pmatrix} 1 \end{pmatrix}$$

In all of the cases, we have $0 \leq det(I - Q) \leq 1$.

Assume the result of the lemma hold for $(n-1) \times (n-1)$ matrix, we want to prove that this result also holds for $n \times n$ matrix. Now, given a $n \times n$ matrix Q, according to the definition in the lemma, we know that the sum of all the entries in Q is at most n. As a result, there exists a column such that the sum of the entries in the column is at most 1. Using equivalent transform, we have that

- The sum of the entries in the 1^{st} column of Q is at most 1.
- The sum of the entries in each row of Q is at most 1.

Now, for the 1^{st} column of $I - Q$, let's continue using the equivalent transform to move all the non-zero entries to the beginning of this column. The possible non-zero entry set of the 1^{st} column of $I - Q$ is

$$\phi, \{\frac{1}{2}\}, \{1\}, \{\frac{1}{2}, -\frac{1}{2}\}, \{1, -\frac{1}{2}\}, \{1, -1\}, \{1, -\frac{1}{2}, -\frac{1}{2}\}$$

The first three cases, the result in the lemma can be easily proved. In the following proof, we only consider the other cases (let C_1 denote the non-zero entry set for the 1^{st} column of $I - Q$) :

(1) $C_1 = \{\frac{1}{2}, -\frac{1}{2}\}$.

In this case, we can write Q as

$$Q = \begin{pmatrix} \frac{1}{2} & A \\ \frac{1}{2} & B \\ O & C \end{pmatrix}$$

where A has at most one non-zero entry $-\frac{1}{2}$, the same as B.

Let

$$E_1 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \end{pmatrix}$$

$$I_1 = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

then we have

$$\begin{aligned} & det(I-Q) \\ = & \frac{1}{2}\det\begin{pmatrix} -A \\ I_1-C \end{pmatrix}+\frac{1}{2}\det\begin{pmatrix} E_1-B \\ I_1-C \end{pmatrix} \\ = & \frac{1}{2}\det\begin{pmatrix} E_1-A-B \\ I_1-C \end{pmatrix} \\ = & \frac{1}{2}\det(I-\begin{pmatrix} A+B \\ C \end{pmatrix}) \end{aligned}$$

Let $D=A+B$, since both A and B has at most one non-zero entry $\frac{1}{2}$, we know that each entry of D is from $\{0,\frac{1}{2},1\}$, and the sum of all the entries is at most one. According to our assumption, we know that

$$0\le\det(I-\begin{pmatrix} D \\ C \end{pmatrix}\le 1$$

As a result, we have

$$0\le\det(I-Q)\le\frac{1}{2}$$

(2) $C_1=\{1,-\frac{1}{2}\}$.

In this case, we can write Q as

$$Q=\begin{pmatrix} 0 & A \\ \frac{1}{2} & B \\ O & C \end{pmatrix}$$

Then

$$\begin{aligned} & det(I-Q) \\ = & \frac{1}{2}\det\begin{pmatrix} -A \\ I_1-C \end{pmatrix}+\det\begin{pmatrix} E_1-B \\ I_1-C \end{pmatrix} \\ = & \frac{1}{2}\det\begin{pmatrix} 2E_1-A-2B \\ I_1-C \end{pmatrix} \\ = & \frac{1}{2}\det(I-\begin{pmatrix} A \\ C \end{pmatrix})+\frac{1}{2}\det(I-\begin{pmatrix} 2B \\ C \end{pmatrix}) \end{aligned}$$

According to our assumption

$$0\le\det(I-\begin{pmatrix} A \\ C \end{pmatrix})\le 1$$

$$0\le\det(I-\begin{pmatrix} 2B \\ C \end{pmatrix})\le 1$$

so $det(I-Q)$ is also bounded by 0 and 1.

(3) $C_1=\{1,-1\}$

Using the same argument as case (1), we can get the result in the lemma.

(4) $C_1=\{1,-\frac{1}{2},-\frac{1}{2}\}$.

In this case, we can write Q as

$$Q=\begin{pmatrix} 0 & A \\ \frac{1}{2} & B \\ \frac{1}{2} & C \\ O & D \end{pmatrix}$$

Let

$$E_2=\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \end{pmatrix}$$

$$I_2=\begin{pmatrix} 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

Then

$$I-Q=\begin{pmatrix} 1 & -A \\ -\frac{1}{2} & E_1-B \\ -\frac{1}{2} & E_2-C \\ O & I_2-D \end{pmatrix}$$

$$\begin{aligned} & det(I-Q) \\ = & \det\begin{pmatrix} E_1-B \\ E_2-C \\ I_2-D \end{pmatrix}+\frac{1}{2}\det\begin{pmatrix} -A \\ E_2-C \\ I_2-D \end{pmatrix} \\ & -\frac{1}{2}\det\begin{pmatrix} -A \\ E_1-B \\ I_2-D \end{pmatrix} \\ = & \frac{1}{2}\det\begin{pmatrix} E_1-B-A \\ E_2-C \\ I_2-D \end{pmatrix}+\frac{1}{2}\det\begin{pmatrix} E_1-B \\ E_2-C-A \\ I_2-D \end{pmatrix} \end{aligned}$$

Now, we can write $A=E+F$ such that both E and F has at most one non-zero entry, which is $\frac{1}{2}$. Therefore,

$$\begin{aligned} & det(I-Q) \\ = & \frac{1}{2}\det\begin{pmatrix} E_1-B-E-F \\ E_2-C \\ I_2-D \end{pmatrix} \\ & +\frac{1}{2}\det\begin{pmatrix} E_1-B \\ E_2-C-E-F \\ I_2-D \end{pmatrix} \end{aligned}$$

where

$$\begin{aligned} & \det\begin{pmatrix} E_1-B-E-F \\ E_2-C \\ I_2-D \end{pmatrix} \\ = & \det\begin{pmatrix} E_1-B-E \\ E_2-C-F \\ I_2-D \end{pmatrix}+\det\begin{pmatrix} -F \\ E_2-C \\ I_2-D \end{pmatrix} \\ & +\det\begin{pmatrix} E_1-B-E \\ F \\ I_2-D \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} & \det\begin{pmatrix} E_1-B \\ E_2-C-E-F \\ I_2-D \end{pmatrix} \\ = & \det\begin{pmatrix} E_1-B-F \\ E_2-C-E \\ I_2-D \end{pmatrix}+\det\begin{pmatrix} E_1-B \\ -F \\ I_2-D \end{pmatrix} \\ & +\det\begin{pmatrix} F \\ E_2-C-E \\ I_2-D \end{pmatrix} \end{aligned}$$

Finally, we can get that

$$\begin{aligned} & det(I-Q) \\ = & \frac{1}{2}\det[I-\begin{pmatrix} B+E \\ C+F \\ D \end{pmatrix}]+\frac{1}{2}\det[I-\begin{pmatrix} B+F \\ C+E \\ D \end{pmatrix}] \end{aligned}$$

According to our assumption, we have that

$$0 \leq \det[I-\begin{pmatrix} B+E \\ C+F \\ D \end{pmatrix}] \leq 1$$

$$0 \leq \det[I-\begin{pmatrix} B+F \\ C+E \\ D \end{pmatrix}] \leq 1$$

Therefore, the result of this lemma holds.

This completes the proof. ■

Lemma 2. *Given a desired probability distribution* $\{q_1, q_2, ..., q_m\}$ *and* $m < 6$*, Huffman procedure can construct a binary-tree such that*

1) *It has* m *leaves with weight* $q_1, q_2, ..., q_m$.
2) $L = \sum_{j=1}^{m-1} \log_2 w_j$ *is minimized, where* w_j *is the weight of* j^{th} *parent node in a binary tree with* m *leaves.*

Proof: It is easy to prove that the case for $m = 3$ or $m = 4$ is true. In the following proof, we only show the case for $m = 5$ briefly. W.l.o.g, we assume $q_1 \leq q_2 \leq ... \leq q_5$. Without considering the order of the leaves, we have only two binary-tree structures, as shown in Fig. 14.

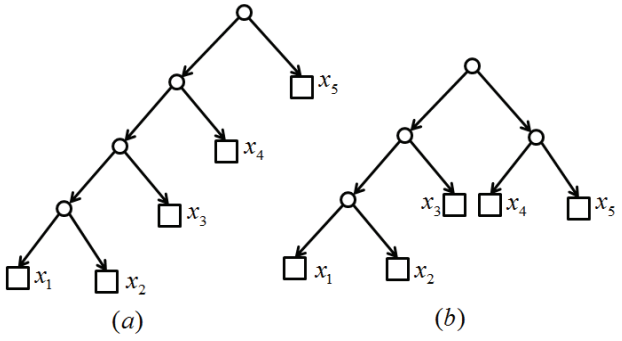


Fig. 14. Two possible tree structures for $m = 5$.

In both of the structures, for any pair of leaves x_i and x_j, if x_i's sibling is x_j's ancestor then $x_i \geq x_j$. Otherwise, we can switch the position of x_i and x_j to reduce $\sum_{j=1}^{m-1} \log_2 w_j$. So if the tree structure (a) in Fig. 14 is the optimal one, we have $x_1 = q_1, x_2 = q_2$ or $x_1 = q_2, x_2 = q_1$. Now, we will show that if the tree structure (b) in Fig. 14 is the optimal one, we also have $x_1 = q_1, x_2 = q_2$ or $x_1 = q_2, x_2 = q_1$.

For the tree structure (b), we have the following relations:

$$x_3 \geq \max\{x_1, x_2\}$$

$$x_4 + x_5 \geq \max\{x_1 + x_2, x_3\}$$

Then q_1 and q_2 is in $\{x_1, x_2, x_4, x_5\}$ and $x_1 + x_2 \leq \frac{1-x_3}{2}$.

Let $x = x_1 + x_2$, then L can be written as

$$\begin{aligned} L &= \min \log(x_1+x_2) + \log(x_1+x_2+x_3) + \log(x_4+x_5) \\ &= \min \log((x_1+x_2)(x_1+x_2+x_3)(1-x_1-x_2-x_3)) \\ &= \min \log x(1-x_3-x)(x+x_3) \end{aligned}$$

So we can minimize $x(1-x_3-x)(x+x_3)$ instead of minimizing L. Fixing x_3, we can see that $x(1-x_3-x)$ increases as x increases when $x \leq \frac{1-x_3}{2}$; $(x+x_3)$ also increases as x increases. So fixing x_3, $x(1-x_3-x)(x+x_3)$ is minimized if and only if x is minimized, which will cause $x_1 = q_1, x_2 = q_2$ or $x_1 = q_2, x_2 = q_1$.

Based on the discussion above, we know that in the optimal tree, q_1 and q_2 must be siblings. Let's replace q_1, q_2 and their parent node using a leaf with weight $q_1 + q_2$. Then we can get an optimal tree for distribution $\{q_1+q_2, q_3, q_4, q_5\}$, whose L value is L_4^*. Assume the optimal L value for distribution $\{q_1, q_2, q_3, q_4, q_5\}$ is L_5^*, then

$$L_5^* = L_4^* + \log_2(q_1 + q_2)$$

Let's consider a tree constructed by Huffman procedure for $\{q_1, q_2, q_3, q_4, q_5\}$, whose L value is L_5. We want to show that this tree is optimal. According to the procedure, we know that q_1 and q_2 are also siblings. By combing q_1 and q_2 to a leaf with $q_1 + q_2$, we can get a new tree. This new tree can be constructed by applying Huffman procedure to distribution $\{q_1 + q_2, q_3, q_4, q_5\}$. Due to our assumption for $m = 4$, it is optimal, as a result the following result is true,

$$L_5 = L_4^* + \log_2(q_1 + q_2)$$

Finally, we can obtain $L_5 = L_5^*$, which shows that the L value of the tree constructed by Huffman procedure is minimized when $m = 5$.

This completes the proof. ■

REFERENCES

[1] J. von Neumann, "Various techniques used in connection with random digits", Appl. Math. Ser., Notes by G.E. Forstyle, Nat. Bur. Stand., vol. 12, pp. 36-38, 1951.

[2] D. Knuth and A. Yao, "The complexity of nonuniform random number generation", in Algorithms and Complexity, New Directions and Results, J.F. Traub, Ed. New York: Academic, pp. 357-428, 1976.

[3] D. Soloveichik, G. Seelig, and E.Winfree, "DNA as a Universal Substrate for Chemical Kinetics", In LNCS 5347, pp. 57-69, 2009.

[4] P. Elias, "The efficient construction of an unbiased random sequences", Ann. Math. Statist., vol. 43, pp. 865-870, 1972.

[5] S. Pae, M.C. Loui, "Optimal Random Number Generation from a Biased Coin", Proceeding of ACM-SIAM symposium on discrete algorithms, pp. 1079-1088, 2005.

[6] T.S. Han, M. Hoshi, "Interval Algorithm for Random Number Generation", IEEE Transaction on Information Theory, vol.43, No.2, pp. 599-611, 1997.

[7] J. Abrahams. "Generation of discrete distributions from biased coins", IEEE Transactions on Information Theory. Vol. 42, pp. 1541-1546, 1996.

[8] M. Blum, "Independent Unbiased Coin Flips From A Correlated Biased Source - A Finite State Markov Chain", Combinatorica, Vol. 6, No. 2, pp. 97-108, 1986.

[9] R. Gill, "On a Weight Distribution Problem, with Application to the Design of Stochastic Generators", Journal of the ACM (JACM), Vol. 10, pp. 110-121, 1963.

[10] D. Wilhelm and J. Bruck, "Stochastic switching circuit synthesis", IEEE International Symposium on Information Theory, pp. 1388-1392, 2008.

[11] W. Qian and M.D. Riedel, "The Synthesis of Combinational Logic to Generate Probabilities", International Conference on Computer-Aided Design, 2009.

[12] M.E. Van Valkenburg, ”Network Analysis”, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ, USA, 1974.
[13] T.M. Cover, J.A. Thomas. ”Elements of Information Theory”, Wiley-Interscience, Second Edition, 2006.